

Loway



QueueMetrics call center suite

The QueueMetrics Unloader User Manual

2017/03/21

The QueueMetrics Uniloader User Manual

Loway

2017/03/21

Revision History

Revision 17.03 - covers Uniloader 2017/03/21

L

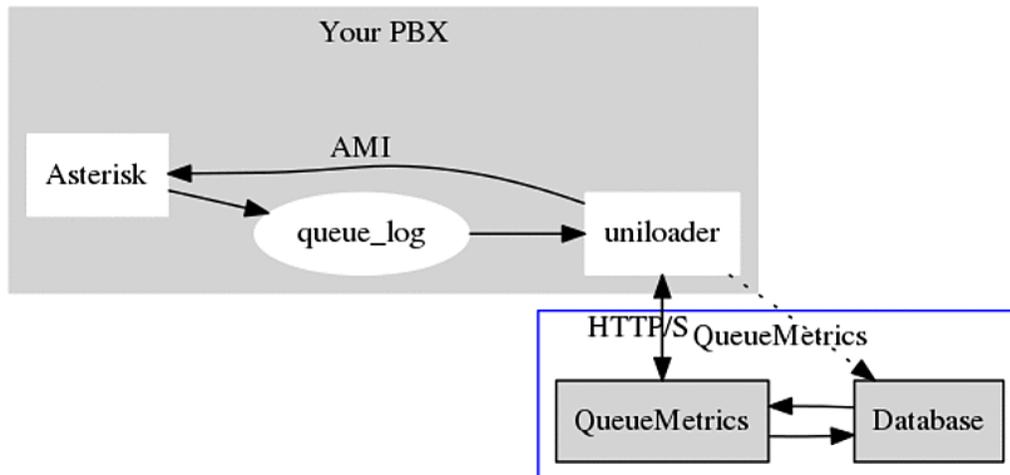
0.3.3

Table of Contents

1. What is Uniloader?	1
What happens if.... ..	1
2. Installation	3
Automated installation (RPM)	3
Manual installation	3
Running in production	4
Example: Uploading data to a local QueueMetrics system	4
Example: Uploading data to a QueueMetrics Live system	4
3. Concepts	6
Back-ends	6
4. Usage	8
Uploading data	8
Feedback actions: proxying AMI	8
Splitting a single queue_log file into multiple back-ends	9
5. Scenarios	12
One Asterisk instance, one local QueueMetrics instance	12
One Asterisk instance, one hosted QueueMetrics Live instance	12
Multiple Asterisk instances, one hosted QueueMetrics-Live instance	12
Multiple Asterisk instances, one QueueMetrics instance	13
One Asterisk instance, multiple QueueMetrics-Live instances	13
Multiple Asterisk instances, multiple QueueMetrics instances	14
6. Additional functions	15
Tracking Music-on-Hold	15
Diagnostics: AMI connection test	16
7. Custom PBX settings	18
Yeastar myPBX	18

Chapter 1. What is Unloader?

Unloader is a program that is installed on your Asterisk PBX. It uploads data to a local or remote QueueMetrics instance and receives actions to be performed on the local PBX.



Unloader is deployed as a single binary file that has to be installed on the PBX itself. It is designed as a very lightweight application so it can work unobtrusively even on low-end hardware; and it is meant to be very safe, so data will not be lost even in cases where the remote QueueMetrics server becomes unavailable.

When it runs, it uploads data using either HTTP/S or the MySQL protocol (depending on the back-end you specify). HTTP/S is meant to run with remote QueueMetrics instances, especially QueueMetrics Live (see <http://queuemetrics-live.com> for more information), while MySQL is meant for local systems; either case works if QueueMetrics is hosted on the same machine.

When running over HTTP, if the QueueMetrics server has no direct connection to the PBX, Unloader is able to act as a proxy and will receive actions to be performed on the Asterisk server via AMI (Asterisk Manager Interface). This way you can run QueueMetrics remotely and still take advantage of the ability to log agents on and off, pause them, listen to calls via ChanSpy, etc.

Unloader is also used to perform other administrative/complementary tasks that perform useful functions on an Asterisk system connected to QueueMetrics; for example, it can generate music-on-hold events on queues, and can help diagnosing issues.

What happens if....

- *The queue_log file is rotated:* Unloader will detect the rotation and will pick up the new file
- *The remote system becomes unavailable:* Unloader will keep on trying until the system is back on line. When it is, it will detect how much data was uploaded and will reload the missing parts. If using multiple remote systems, only data for the specific remote system will be blocked, and other systems will keep on working in real-time.
- *Asterisk becomes unavailable:* any commands meant to be sent to Asterisk are queued and will be performed when Asterisk comes back on-line

-
- *The PBX reboots*: you need to make sure Uniloader starts on boot. It has no service dependencies so it can be started at any time.

Chapter 2. Installation

If you run a CentOS-based PBX system, you can use the easier RPM install; if not refer to the manual install below.

Automated installation (RPM)

You can just run the following commands on your PBX:

```
wget -P /etc/yum.repos.d http://yum.loway.ch/loway.repo
yum install uniloader
```

Uniloader will be installed under `/usr/local/uniloader` and will be added to the system path. A basic configuration will be set in `/etc/sysconfig/uniloader`; Uniloader will be immediately started as a daemon and it will start automatically on reboot.

You will be able to control Uniloader by entering:

```
service uniloader start
service uniloader stop
service uniloader restart
```

Manual installation

The Uniloader can be downloaded from <http://queuemetrics-live.com/uniloader.jsp>

The package contains:

- Uniloader binaries for all supported architectures (i386, amd64, arm7),
- A sample `extensions_queuemetrics` file,
- A sample splitter file.

Just copy the file "uniloader_xxx" for your architecture (Intel 32 / 64 bit, ARM 7) into your computer and make it executable:

```
cp ./bin/uniloader_arm7 ./uniloader
chmod +x ./uniloader
```

To test it, run:

```
./uniloader -?
```

It should output a result like:

```
NAME:
  uniloader - uploads a queue_log file to QueueMetrics or QueueMetrics Live.
USAGE:
  uniloader [global options] command [command options] [arguments...]
VERSION:
  0.3.3 - build: - OS: darwin/amd64 - RT: go1.7
COMMANDS:
  upload, u   Uploads a source file to a QueueMetrics instance
  trackmoh, m Tracks Music-on-Hold on queues.
  amitest     Tests an AMI connection
  help, h     Shows a list of commands or help for one command
GLOBAL OPTIONS:
  --src, -s "/var/log/asterisk/queue_log"  The source queue_log file to be uploaded
```

```

--cacert                An optional CA Cert file, in .pem format
--verbose-back-end      Enables verbose back-end logging
--read-pipe             The source file is a pipe
--help, -h             show help
--version, -v          print the version

```

If it does, it is working.

Running in production

Uniloader produces a verbose log on STDOUT that should be redirected to a file and periodically rotated.

You should also make sure that Uniloader is started when the PBX boots and that in case it should crash it is automatically restarted.

We advise running Uniloader using *nice* so that it has reduced access to scarce CPU resources in case of high load / contention with the PBX - while the PBX voice quality quickly degrades on a resource-starved system, Uniloader does not really care about small delays in data uploading.

Example: Uploading data to a local QueueMetrics system

This is the most common scenario when using a locally installed QueueMetrics system.

```

nohup nice \
./uniloader -s /var/log/asterisk/queue_log \
    upload --uri "mysql:tcp(1.2.3.4:3306)/queuemetrics?allowOldPasswords=1" \
        --login queuemetrics --pass javadude --token P001 \
    >> /var/log/uniloader.log &

```

Will upload the queue_log file located at /var/log/asterisk/queue_log to the remote "queuemetrics" database on server 1.2.3.4 with login "queuemetrics" and password "javadude", using the default partition "P001".

After you start it, check the file /var/log/uniloader.log to make sure there are no errors. The most common error is that you did not create the correct grants for your MySQL user to upload data remotely.

If everything seems to work, log in into QueueMetrics, select "Edit system parameters" and make sure that the default partition is P001.

```

# This is the default queue log file.
default.queue_log_file=sql:P001

```

At this point, log off; log on again and click on "Mysql storage information"; select partition P001 and select "Autoconfigure queues". Now the default queue "00 All" will include all of your queues and you can see your historical and real-time status.

See also One Asterisk instance, one local QueueMetrics instance.

Example: Uploading data to a QueueMetrics Live system

You received your QueueMetrics Live access information; and in the received email you find that your instance is called "ABCD" and the password is "1234".

```
nohup nice \  
./uniloader --src=/var/log/asterisk/queue_log \  
  upload --uri https://my.queuemetrics-live.com/ABCD \  
  --login webqloader --pass 1234 \  
>> /var/log/uniloader.log &
```

At this point Uniloader will start feeding the remote database. You can login at any time by visting the address <https://my.queuemetrics-live.com/ABCD> and logging in as "demoadmin" and the password you were given.

The first time you log in, click on "System diagnostic tools" and then "Live DB inspector" to see data being uploaded. When your database is complete (this may take a few minutes, depending on how much data is on your PBX), go back to the home page and click on "Mysql storage information"; select partition P001 and select "Autoconfigure queues". Now the default queue "00 All" will include all of your queues and you can see your historical and real-time status.

See also One Asterisk instance, one hosted QueueMetrics Live instance. If you run a Yeastar system, see the chapter dedicated to Yeastar MyPBX the section called "Yeastar myPBX" [18].

Chapter 3. Concepts

Back-ends

Uniloader supports three different back-ends: HTTP/HTTPS, MYSQL and FILE.

Each back end is functionally similar and can be thought of as a black box; it can be selected simply by entering a proper URI for the server.

HTTP/HTTPS back-end

If your URI looks like:

```
http://myserver/queuemetrics
```

Then you are using HTTP. In this case, the value of the "token" parameter is either a server in a cluster, or you can leave it blank to denote the default server, and user/password are for a valid QueueMetrics HTTP user.

The HTTP back-end also supports HTTPS URLs and will, by default, retrieve actions to be performed on the PBX.

Please note that some appliances do not support HTTPS, so running HTTP might be mandatory.

HTTPS CA certificate issues

On some systems (especially appliances) it is possible that when running HTTPS requests, they all fail with an error like:

```
x509: failed to load system roots and no roots provided
```

In this case, you have to manually tell the Uniloader where to find the correct CA .pem files for your system, by using the "--cacert" parameter.

E.g.

```
./uniloader --cacert=/etc/certs/default.pem upload ....
```

Will force Uniloader to use the supplied root certificates. In case they are totally missing, we suggest copying a recent certificate file from a working Linux distribution and point to that.

MySQL back-end

If your URI looks like:

```
mysql:127.0.0.1/queuemetrics
```

the loader will connect to a MySQL database called "queuemetrics" on "127.0.0.1", using the supplied login and password; the token in this case is the partition that we want to upload data to.

If your MySQL is running on a remote system, it might be advisable to use a MySQL URI of the format:

```
mysql:tcp(1.2.3.4:3306)/uniloader?allowOldPasswords=1
```

This will connect to a database called "uniloader" on 1.2.3.4 and will set the parameter "allowOldPasswords" to 1, as it is sometimes needed to use old versions of MySQL.



A complete reference of all allowed DSN (Data Source Name) formats and connection parameters is available at <https://github.com/go-sql-driver/mysql>

File back-end

If your URI looks like:

```
file:/my/file/path
```

The loader will try and append to a local file. This module is meant for quick testing of splitting rules and does not currently check the state of the local file before writing to it.

It can also be used as a quick way to "throw away" a log file, by using *file:/dev/null* on Unix systems.



This back-end is only meant for testing and experimentation; the file is rebuilt on every run, so no guarantee about data integrity is implied.

Chapter 4. Usage

Uploading data

To upload data, you need the upload command in Uniloader:

```
NAME:
  upload - uploads a source file to a QueueMetrics instance

USAGE:
  command upload [command options] [arguments...]

OPTIONS:
  --uri, -u                The connection URI. Valid URIs start with file:, mysql:, http:, https:
  --login, -l "webqloader" The login for your connection
  --pass, -p "qloader"     The password for your connection
  --token, -t              In MySQL mode, the partition. In HTTP/S mode, usually blank or server-id
  --splitter, -x          A JSON file describing how to split the source into multiple QM instances
  --noActions              Actions from QM will NOT be sent to the PBX via AMI. Requires HTTP/S.
```

So you usually launch it like:

```
./uniloader --src /var/log/asterisk/queue_log upload \
  --uri mysql://queuemetrics --login qm --pass 1234 --token P001
```

You can avoid passing parameters which value matches the defaults, so if your token is blank, or your user is "webqloader" (as it is the case with default QueueMetrics Live instances), you do not need to pass them explicitly.

Uniloader reads the source file specified in "src" and automatically detects if the file is rotated/rewritten.

When data is being uploaded, Uniloader makes sure that data is not uploaded twice and retries on errors. You can safely restart it at any time and it will automatically synchronize with the current state of the selected back-end.



You can NEVER have multiple Uniloader / Qloader / Wqloader instances point to the same instance at the same time. If you do, you will get hard-to-debug data corruption.

Feedback actions: proxying AMI

It is possible for Uniloader to act as a kind of AMI (Asterisk Manager Interface) proxy for a remote QueueMetrics instance. This happens by default if you use a HTTP back-end. If you do not want this feature, you need to start Uniloader with the "--noActions" option.

For example:

```
./uniloader --src /var/log/asterisk/queue_log upload \
  --uri http://my.queuemetrics-live.com/test1234 --pass 1234
```

All access information to the Asterisk PBX is to be configured on the QueueMetrics instance; for example, if the PBX server is accessible on the address 127.0.0.1 (so the same host Uniloader is running on) and you log-in as "admin" password "amp123", you should edit the configuration properties and make sure that it says:

```
callfile.dir=tcg:admin:amp123@127.0.0.1
default.webloaderpbx=true
```

At the PBX level, make sure you include the default QueueMetrics dial-plan in extensions.conf:

```
#include extensions_queuemetrics.conf
```

And reload it.

Splitting a single queue_log file into multiple back-ends

If you run a single Asterisk instance on which multiple clients are hosted, chances are that you configure your Asterisk system with a common naming convention, so that all extensions for your client Foo Company are named "foo-123", all queues are named "foo-q1" and so on.

If you do, it is actually possible to split the queue_log file that Asterisk generates into multiple virtual queue_log files. To do this, Uniloader looks for references of the client name in queues and agents, and can optionally rewrite them so that a reference for queue "foo-q1" is sent to a specific QueueMetrics Live instance set up just for Foo Company; and it is rewritten as simply "q1".

To split a single queue_log file you need to create a split file that details what you want done, and then you can launch:

```
./uniloader --src queue_log.txt upload --splitter splitter.json
```

Please note that you do not need to specify a "main" rule on the command line. If you do, a copy of the source file will be also uploaded to the main driver, without applying any transformation.

These are sample contents for a splitter.json file:

```
[
  {
    "uri": "http://my.queuemetrics-live.com/foocompany",
    "login": "webqloader",
    "pass": "verysecure",
    "token": "",
    "matcher": ["foo-"],
    "match": "any",
    "removematch": true,
    "disabled": false,
    "noactions": false,
    "clientname": "foo"
  },
  {
    "uri": "mysql:127.0.0.1/queuemetrics",
    "login": "queuemetrics",
    "pass": "itsasecret",
    "token": "P001",
    "matcher": ["bar-"],
    "match": "any",
    "removematch": false
  }
]
```

The following items must be specified for each instance.

- uri: the URI to upload data to. You can mix and match different backends as you see fit
- login, pass and token: the information required by your back-end
- matcher: an array of strings that will be searched in the agent and queue fields.
- match: at the moment, it must be "any" - meaning that if a string is found, it is considered a match
- removematch: if true, the matching string is removed from the queue and agent fields
- disabled: set to true to manually turn off a rule

- `noactions`: set to true to turn off AMI actions for this instance, as you would do for the main instance by using the "--noActions" flag.
- `clientname`: the name of the instance, that will be injected in the AMI responses using the dialplan variable `UNILoader_CLIENT` before they are passed to Asterisk. It will also be used to replace the sequence `!UNILoader_CLIENT` in your Asterisk channels.

If you avoid setting some item, it is assumed to be a blank string or the "false" boolean value. Defaults you set with the command lines are ignored, so all relevant information must be specified in the JSON file.



Split data is sent only to instances matching the specific split rule; so the main instance you specify on the command line will be fed all data in any case. As you usually do not want this, you can simply avoid entering any "--uri" parameter on the command line.

Splitting FAQs

What happens if one back-end is or becomes unavailable?

Each back end runs in parallel; but if one should lag behind or should not be available, data for it is delayed until the system is fully operational; at that point it will catch up automatically.

You can also safely restart Uniloder even if not all data is currently uploaded to all instances; the only thing you have to consider is that, in case your `queue_log` is rotated, then only data present in the current `queue_log` file is uploaded.



This works correctly only for the MySQL and HTTP drivers; in case you specify a file back end, it will be truncated and rebuilt on each invocation.

Can I use different back-ends?

Yes, of course. Mix and match them as you best see fit.

Can I use feedback actions?

Yes - provided that all back-ends are HTTP.

What happens to the default back-end?

The default back-end - the one that is specified on the command line - is sent the raw `queue_log` data. If you don't need this, you can use a file back-end and point it to `/dev/null`, or you can simply omit it.

Do I have to have a splitting rule for all my virtual clients?

No. Only the rules you specify will be applied, so if you do not include a rule for a specific client, the relevant logs will simply be ignored. This means that you may host on the same Asterisk instance clients who use QueueMetrics and clients that don't.

How do I modify the configuration on a live system?

You can simply create a new JSON file and restart the Uniloder. In a few seconds it will sync again and start tailing the files. The file will be read in parallel by all the different back-ends, so it will not require a proportional amount of disk IOPS.

Why do I need the clientname field?

If you have a scenario where multiple QM instances are fed by the main QueueMetrics instance, it will be handy to have rewriting enabled, so that e.g. the queue called "foo-q1" appears at the QueueMetrics level as simply "q1".

This works fine when uploading data to QueueMetrics, but when actions are performed by that QueueMetrics instance, they will appear as happening on queue "q1" and not on the actual Asterisk queue "foo-q1".

By injecting the variable UNILoader_CLIENT is therefore possible to edit the actions dialplan and rebuild the correct physical name to be used when performing actions at the Asterisk level.

Chapter 5. Scenarios

One Asterisk instance, one local QueueMetrics instance

You want to use Uniloader for your local QueueMetrics instance.

In this case you should use the MySQL back-end. You would not usually use the AMI feedback as QueueMetrics is able to connect directly to the PBX.

You may also use the HTTP back-end, but there are currently no advantages in doing so.

See also: Example: Uploading data to a local QueueMetrics system the section called “Example: Uploading data to a local QueueMetrics system” [4].

One Asterisk instance, one hosted QueueMetrics Live instance

You want to use Uniloader for a QueueMetrics-Live instance.

In this case you should use the HTTP or HTTPS backend, and turn on AMI feedback, as the QueueMetrics Live instance has no way to connect directly to your PBX.

In order to make your life easier, QueueMetrics Live actions are pre-configured to send actions back via HTTP; you just need to make sure that the AMI credentials specified in the property *callfile.dir* in *configuration.properties* match the ones used on your PBX.

See also: Example: Uploading data to a QueueMetrics Live system the section called “Example: Uploading data to a QueueMetrics Live system” [4].

Multiple Asterisk instances, one hosted QueueMetrics-Live instance

You have multiple Asterisk boxes and want to consolidate all their activity into a single QueueMetrics-Live instance.

QueueMetrics-Live supports up to 5 Asterisk instances in a cluster for a total of 50 agents per instance. Each Uniloader instance must upload data to a different partition. Each cluster member should be defined in the QueueMetrics-Live instance.

For example, this is how you would configure three Asterisk instances on a QueueMetrics-Live system:

```
default.queue_log_file=cluster:*
cluster.servers=srva|srvb|srvc
cluster.srva.manager=tcp:dial:12345@127.0.0.1
cluster.srva.queue_log=sql:A
cluster.srvb.manager=tcp:dial:12345@127.0.0.1
cluster.srvb.queue_log=sql:B
```

```
cluster.srvc.manager=tcp:dial:12345@127.0.0.1
cluster.srvc.queuelog=sql:C
```

In this case, you would run Uniloader with the token "srva" on server A, and it would upload data to partition A:

```
./uniloader --src=/var/log/asterisk/queue_log
            upload --uri https://my.queuemetrics-live.com/MYINSTANCE
            --login webqloader --pass CHANGEME --token=srva
```

The same goes for servers "srvb" and "srcv" that would upload to B and C respectively.

As you can see, each cluster member in QueueMetrics defines its own AMI credentials; so you can safely use the AMI feedback mode with no further configuration. Please note that it's common for all AMI instances to just point to "127.0.0.1" because Uniloader runs on the same box as Asterisk itself.



When configuring agents, make sure that you set the server for each agent, so when they log in to queues they already point to the right server.

Multiple Asterisk instances, one QueueMetrics instance

You have multiple Asterisk boxes and want to consolidate all their activity into a single QueueMetrics instance.

In this case, you need a cluster-enabled QueueMetrics instance, and each Uniloader instance should upload data to a different partition. Each cluster member should be defined in the QueueMetrics instance (if you use the MySQL back-end, you set the token to the name of the partition; if you use HTTP you should set it to the name of the cluster member).



The names for cluster members are the ones you use in the property *cluster.servers* of your QueueMetrics *configuration.properties* file.

Each cluster member in QueueMetrics defines its own AMI credentials; so you can safely use the AMI feedback mode with no further configuration.

One Asterisk instance, multiple QueueMetrics-Live instances

You run multiple different clients on one Asterisk instance, and you want to send each of them to their own QueueMetrics Live instance.

In this case, you need to set up splitting rules so that data for each client is uploaded to the right QueueMetrics Live instance.

For example, your *'splitter.json'* file could look like:

```
[
  {
    "uri": "http://my.queuemetrics-live.com/client3",
    "login": "webqloader",
    "pass": "CHANGEME",
    "token": ""
```

```

        "matcher": ["client3-"],
        "match": "any",
        "removematch": true,
        "disabled": false,
        "noactions": false,
        "clientname": "client3-"
    },
    ...other clients...
]

```

In order to make sure that Asterisk performs the correct actions at the AMI level, you must specify a "clientname" for each client and use that string in the Asterisk dialplan (where it is returned under the variable "UNILoader_CLIENT") in order to build the actual queue / agent / channel name to be used on Asterisk.

So you would edit the stanzas you want to use in *extensions_queuemetrics.conf* to use the client name, like e.g.:

```

; extension 37: agent removequeuemember with hotdesking (for asterisk v1.4+)
exten => 37,1,Answer
exten => 37,2,NoOp( "QM: RemoveQueueMember (asterisk v1.4+) Agent/${AGENTCODE}
                 at extension SIP/${QM_AGENT_LOGEXT} on queue ${QUEUENAME}
                 made by '${QM_LOGIN}' for '${UNILoader_CLIENT}'" )
exten => 37,3,RemoveQueueMember(${UNILoader_CLIENT}${QUEUENAME},SIP/${UNILoader_CLIENT}${QM_AGENT_LOGEXT})
exten => 37,4,Hangup

```

So if this action is performed on "client3" removing extension "127" from queue "300", the actual action performed would be:

```
RemoveQueueMember(client3-300,SIP/client3-127)
```

That would produce a queue_log record like:

```
1487239051|1487239051.123|client3-300|SIP/client3-127|REMOVEMEMBER
```

But the splitter would then upload it to QueueMetrics as if it was:

```
1487239051|1487239051.123|300|SIP/127|REMOVEMEMBER
```

Because it would match the string "client3-" in both the queue and agent fields. This way each QueueMetrics-Live instance is blissfully unaware of the physical names for queues and agents that are used at the Asterisk level.

Also, as for some actions (chanspy and originate) QueueMetrics need to originate calls directly within your dialplan, you should edit the *configuration.properties* file so that channels where the client is required appear as:

```

callfile.monitoring.channel=SIP/$EM-!UNILoader_CLIENT
callfile.outmonitoring.channel=SIP/$EM-!UNILoader_CLIENT
callfile.customdial.channel=SIP/$EM-!UNILoader_CLIENT

```

Multiple Asterisk instances, multiple QueueMetrics instances

If you have multiple Asterisk instances on which calls are processed, and calls for any client can be processed on each cluster member, you need to set up rewriting rules and create a cluster member (and related partition) on each destination QueueMetrics instance.

Make sure you use the "clientname" variable to have Asterisk perform the correct AMI calls.

Chapter 6. Additional functions

Tracking Music-on-Hold



This feature is experimental.

Unloader can be used to detect and generate Music-on-Hold events for calls that are being handled on Asterisk queues. This works by opening a stream of events from the Asterisk system through AMI and tracking call progress.

```
$ ./unloader trackmoh -?
NAME:
  unloader trackmoh - Tracks Music-on-Hold on queues.
USAGE:
  unloader trackmoh [command options] [arguments...]
OPTIONS:
  --host "127.0.0.1"   Your Asterisk server
  --port "5038"       The AMI port on Asterisk
  --login             The AMI user as defined in manager.conf
  --secret            The AMI secret
  --debugfile        A debug file to dump AMI data to
```

In order to use it, you can launch it as:

```
./unloader trackmoh --login admin --secret amp123
```

Where *admin* and *amp123* are the current AMI credentials for your local Asterisk system. At this point, you should:

- Send a call to an Asterisk queue
- Have an agent handle the call
- Have the agent start and stop music-on-hold

The event should appear in QueueMetrics on the real-time page.

Please note that:

- If you restart the process, the current state will be lost - so Unloader will not be aware on whether a call is on a queue.
- You should run this in parallel to the main data upload

Debugging missed events

If you find that some calls are missing events or have multiple events and you are able to find a consistent pattern, you should run:

```
./unloader trackmoh --login admin --secret amp123 --debugfile debugdata.txt
```

When running in debug mode, a large file is quickly generated; so it is appropriate to run it only for short periods.

When you are done, you should send Loway:

- The file that was just created
- Your queue_log file
- And indication of which call is displaying wrong events

Diagnostics: AMI connection test

Unloader lets you test an AMI port from the command line. It will also check that the *queuemetrics* context is present on the system, and will make sure that the AMI user has the required "originate" privilege.

```
$ unloader amitest -?

NAME:
  unloader amitest - Tests an AMI connection

USAGE:
  unloader amitest [command options] [arguments...]

DESCRIPTION:
  This command test an AMI connection.

  It checks that the `queuemetrics` context is present and its
  functions are present. It checks originates to `10@queuemetrics`
  and prints available queues.

OPTIONS:
  --host "127.0.0.1"  Your Asterisk server
  --port "5038"       The AMI port on Asterisk
  --login             The AMI user as defined in manager.conf
  --secret            The AMI secret
```

In order to use it, you can call it like:

```
$ unloader amitest --login admin --secret amp111
```

It will print out a comprehensive report, like:

```
Testing AMI connection to 10.10.5.27:5038 - Username 'admin' secret '*****'
AMI Connected: Asterisk Call Manager/1.3
```

N.	Meaning	ext@queuemetrics	Present?
1	Dummy extension	10	Ok
2	Chanspy - inbound calls	11	Ok
3	Sets a call status	12	Ok
4	Chanspy - outbound calls	14	Ok
5	Add call feature	16	Ok
6	Remove call feature	17	Ok
7	Agent pause	22	Ok
8	Agent unpaue	23	Ok
9	AddMember	25	Ok
10	RemoveMember	26	Ok
11	Custom dial	28	Ok
# 12	Send SMS to SIP device	29	MISSING
13	Soft hangup of call in queue	30	Ok
14	Redirect call in queue	31	Ok
15	Agent pause with hotdesking	32	Ok
16	Agent unpaue with hotdesking	33	Ok
17	Addmember with hotdesking	35	Ok
18	Removemeber with hotdesking	37	Ok

Known Queues	Completed	Abandoned
- 300	10	5
- 301	0	0
- default	0	0
- 400	0	0

Originate on 10@queuemetrics worked.

This shows:

- The version of AMI in use
- Whether all default queuemetrics extensions are present, and which ones are missing
- The queues configured in Asterisk, and their current usage statistics
- Whether the user has "originate" privileges

If connection is possible, it returns with a status code of zero; if not possible, or wrong credentials are used, it returns with an error code so that you can script it.

Chapter 7. Custom PBX settings

Uniloader can run on several Asterisk-based PBXs - on custom hardware, or in specific distributions. Here are reported some configuration hints for specific systems.

Yeastar myPBX

MyPBX related setup

For this system you need to download the Uniloader Installation script and run it, it will install Uniloader automatically.

- Go into */persistent* in case you have a Yeastar U PBX or */ysdisk/support/tmp* if you have a Yeastar S PBX
- Download the script with *wget http://get.queuemetrics-live.com/yeastar*
- Execute the script with *sh yeastar*
- Follow the instructions
- When finished restart the PBX

QueueMetrics Live related setup

Integrating MyPBX with QueueMetrics Live requires some modifications on the QueueMetrics Live settings. This can be easily performed through a web page tool reachable from the QueueMetrics Live home page by following the below steps:

- Log on QueueMetrics Live with administrative rights
- Click on the "Edit system parameters" under the "Administrative tools" subset
- Look at the configuration key *callfile.monitoring.channel=Local/\$EM@from-internal* and change as *callfile.monitoring.channel=SIP/\$EM*
- Repeat the same for the configuration keys *callfile.outmonitoring.channel* and *callfile.customdial.channel*
- Save, then log-off from QueueMetrics Live